Table of Contents

How to measure the trigger efficiency

This is valid from DaVinci v26r3

Introduction

This page explains the best ways to determine if an event has passed the Hlt, and how.

Why not use Hlt1Global and Hlt2Global

The Hlt flow is set up such that all events with Hlt1Global=1 are passed to Hlt2 and all events with Hlt2Global=1 go to storage. Hence in the real data you will only see events with Hlt1Global=1 and Hlt2Global=1. So why not ask the same for MC?

This simply won't work. Hlt1 and Hlt2 contain a lot of lines that do nothing but help the debugging and the monitoring of the data. For instance all events that go to the express stream (=prompt reconstruction of a few Hz) are in there. You will never be able to take that into account in MC. In the MC the concept of "rate" does not apply. How do we throttle a line to a rate of 10 Hz? Compared to what?

On MC10 we have run some TCK in tagging mode (i.e. don't reject events), but have set all rate limited lines to 0. In the future (as of 24/1/11) we will run a "mirror" TCK used only for MC that also sets all prescales to 1. This was not done on the MC10 campaign. Thanks to the ones who noticed.

In principle all you care about is the efficiency of the **physics** lines. There are two scenarios:

- Either you care about trigger biases (most likely) and therefore you want to concentrate on some well defined lines. In that case look at the appropriate lines.
- Or you don't care and are happy to take into account all lines. Look at the trigger bits. This is not really encouraged. Better know what you're doing.

Hlt Lines

This is the simple situation. You just want to know if you have passed a given line, or a given set of lines. Suppose you want to know what `Hlt2TopoOSTF2BodyDecision` has decided.

- **In an nTuple :** In DecayTreeTuple or EventTuple fill TupleToolTrigger with option `VerboseHlt2 = True`.

```
tuple.addTupleTool("TupleToolTrigger/TTT")
tuple.TTT.VerboseHlt1 = True
tuple.TTT.VerboseHlt2 = True
tlist = [ "Hlt1TrackAllL0Decision", "Hlt2TopoOSTF2BodyDecision" ] # add anything you like.
tuple.TTT.TriggerList = tlist
```

- **In C++** :

```
bool found = false ;
if( exist<LHCb::HltDecReports>( LHCb::HltDecReportsLocation::Default ) ){
  const LHCb::HltDecReports* decReports = get<LHCb::HltDecReports>( LHCb::HltDecReportsLocation::
  for(LHCb::HltDecReports::Container::const_iterator it=decReports->begin(); it!=decReports->end(
    if ( "Hlt2TopoOSTF2BodyDecision" == it->first ) found = it->second.decision() ;   // passed ?
  }
} else Warning("No HltDecReports at "+LHCb::HltDecReportsLocation::Default,StatusCode::FAILURE,1)
```

The `HltDecReports` will be written into the DST. Make sure you also have them on your MicroDst.

- **In a filter**:

```
from Configurables import LoKi__HDRFilter   as HltFilter
MySequencer.Members += [ HltFilter( 'HltPassFilter', Code="HLT_PASS('Hlt2TopoOSTF2BodyDecision')"
```

Note that under 'Code' you can use `HLT_PASS` which expects the name of an Hlt line, or you can use eg. `"HLT_PASS_RE('Hlt1.*Hadron.*Decision')"` which passes if there is at least one line matching the regular expression `Hlt1.*Hadron.*Decision` which accepted the event. Note that you can also do boolean AND and OR, eg

```
Code ="HLT_PASS('Hlt2UnbiasedJPsiDecision') & !HLT_PASS('Hlt2IncMuTrack4JpsiDecision')"
```

would give you events passing `Hlt2UnbiasedJPsiDecision`, but not passing `Hlt2IncMuTrack4JpsiDecision` (provided such events are in your input).

Then there is the equivalent for L0DU, and for ODIN.

```
from Configurables import LoKi__L0Filter as L0Filter
filter = L0Filter('myName', Code = "L0_CHANNEL('Hadron')" )
```

requires the L0 'Hadron' channel to have accepted the event, and again you can do logical AND and OR like in the Hlt case. In addition there is `L0_DECISION` which just checks for the overall L0 decision.

- **In a stripping line**:

The filter is already declared to `StrippingLine`. You can do

```
MBMicroBiasLine  = StrippingLine( "MBMicroBias"
                                  , HLT = "HLT_PASS_RE('Hlt1MBMicro.*Decision')"
                                  , checkPV = False
                                  , prescale = 1
                                  , postscale = 1)
```

- **In gaudipython**:

```
HltReport = evt['Hlt/DecReports']
If not HltReport.decReport('Hlt1MBMicroBiasRZVeloDecision').decision() : continue
```

But what are the best lines on my signal?

Run `ReadHltReport` (ideally at the end of the sequence running your offline selection). It will print out statistics for all trigger lines. Most likely the sequence `Hlt1TrackAllL0Decision` `Hlt2TopoOSTF=n=BodyDecision` with n=1,2,3 will win. (see also DaVinciFAQ)

Routing (aka Trigger) bits

The trigger sets a certain number of bits according to what should happen to the events after the processing in the event filter farm. These bits are set using the HltFilter described above. According to the bit set, the event goes into the monitoring farm, or the express stream. They all go into the main stream. The list is to be found in method `configureRoutingBits` in HltConf.Configuration. They are printed out at the end of the trigger running:

```
      HltRoutingBitsWriter                                    SUCCESS Number of counters : 27
       |    Counter                                 |     #     |    sum     | mean/eff^* |
```

```
| "00:( ODIN_BXTYP == LHCb.ODIN.Beam1 ) | ( ODIN_B|       200 |           0 |     0.0000 |
| "01:( ODIN_BXTYP == LHCb.ODIN.Beam2 ) | ( ODIN_B|       200 |           0 |     0.0000 |
| "08:L0_DECISION_PHYSICS"                        |       200 |         144 |    0.72000 |
| "09:L0_CHANNEL_RE('B?gas')"                     |       200 |           0 |     0.0000 |
| "10:L0_CHANNEL('CALO')|L0_CHANNEL('MUON,minbias'|       200 |           0 |     0.0000 |
| "11:L0_CHANNEL('Photon')|L0_CHANNEL('Hadron')|L0|       200 |         138 |    0.69000 |
|*"32:HLT_PASS('Hlt1Global')"                     |       200 |         144 |(  72.0000 +-
|*"33:HLT_PASS_SUBSTR('Hlt1Lumi')"                |       200 |           0 |(  0.00000 +-
|*"34:HLT_PASS_RE('Hlt1(?!Lumi).*Decision')"      |       200 |         144 |(  72.0000 +-
|*"35:HLT_PASS_SUBSTR('Hlt1Velo')"                |       200 |           0 |(  0.00000 +-
|*"36:scale(HLT_PASS_RE('Hlt2Express.*Decision')|H|       200 |           1 |( 0.500000 +-
|*"37:HLT_PASS('Hlt1ODINPhysicsDecision')"        |       200 |           0 |(  0.00000 +-
|*"38:HLT_PASS('Hlt1ODINTechnicalDecision')"      |       200 |           0 |(  0.00000 +-
|*"39:HLT_PASS_SUBSTR('Hlt1L0')"                  |       200 |         144 |(  72.0000 +-
|*"40:HLT_PASS_RE('Hlt1.*Hadron.*Decision')"      |       200 |          19 |(  9.50000 +-
|*"41:HLT_PASS_RE('Hlt1.*SingleMuon.*Decision')"  |       200 |          22 |(  11.0000 +-
|*"42:HLT_PASS_RE('Hlt1.*DiMuon.*Decision')"      |       200 |          25 |(  12.5000 +-
|*"43:HLT_PASS_RE('Hlt1.*MuTrack.*Decision')"     |       200 |          18 |(  9.00000 +-
|*"44:HLT_PASS_RE('Hlt1.*Electron.*Decision')"    |       200 |           7 |(  3.50000 +-
|*"45:HLT_PASS_RE('Hlt1.*Pho.*Decision')"         |       200 |           3 |(  1.50000 +-
|*"46:HLT_PASS_RE('Hlt1(?!ODIN)(?!L0)(?!Lumi)(?!Te|       200 |          61 |(  30.5000 +-
|*"47:HLT_PASS_RE('Hlt1MBMicroBias.*Decision')"   |       200 |           0 |(  0.00000 +-
|*"48:HLT_PASS('Hlt1MBNoBiasDecision')"           |       200 |           0 |(  0.00000 +-
|*"49:HLT_PASS_SUBSTR('Hlt1BeamGas')"             |       200 |           0 |(  0.00000 +-
|*"64:HLT_PASS('Hlt2Global')"                     |       200 |          34 |(  17.0000 +-
|*"65:HLT_PASS('Hlt2DebugEventDecision')"         |       200 |           0 |(  0.00000 +-
|*"66:HLT_PASS_RE('Hlt2(?!Transparent).*Decision')|       200 |          34 |(  17.0000 +-
```

Bits 46 (Hlt1 physics) and 66 (Hlt2 physics) are the most interesting.

- Hence you can get them by using the same pattern in an HltFilter.
- You also have them in your nTuple, as above.
- Finally you can read them in C++ from the raw event:

```
#include "Kernel/ReadRoutingBits.h"  // in HltInterfaces

  if (exist<LHCb::RawEvent>(LHCb::RawEventLocation::Default)){
    LHCb::RawEvent* rawEvent = get<LHCb::RawEvent>(LHCb::RawEventLocation::Default);
    std::vector<unsigned int> yes = Hlt::firedRoutingBits(rawEvent);
  }
```

`yes` will be a vector of all bits that have fired. For instance you can search for 66. **Warning**: Do this only on events that pass L0, or the result might be unpredictable.

-- PatrickSKoppenburg - 25-Nov-2009 - 24-Jan-2011

--- This topic: LHCb > HltEfficiency
Topic revision: r14 - 2014-05-22 - OliverGruenberg